Parallel Implementation of a Deep Belief Network

Eric Mullen

Final Project, Neural Networks

Fall 2010

Problem

Deep belief networks are slow

- Training weights between 2 layers of N neurons each using contrastive divergence is a series of O(N^2) operations.
- This leads to training of these networks sometimes being an overnight task.
- RBM-Provisor is awesome, but can take a long time to train.

Training Algorithm

3 distinct parts of training:

- Activate nodes (either hidden -> visible or visible -> hidden)
- Accumulate differences
  - difference between vis node i and hidden node j is the product of the values of the nodes
  - this is added to the weight i,j
- Update Weights
  - actually add the differences to the weights (and multiply by the learning rate in there)

Training Algorithm, cont

To train the weights between two adjacent layers in a deep belief network:

1. Propagate the input from the lowermost I/O nodes into the network, to the particular set of visible nodes you're concerned with
2. Activate the hidden nodes with respect to your visible nodes
3. Accumulate the difference between visible and hidden nodes, call this dPos
4. Activate visible nodes
5. Activate hidden nodes
6. Accumulate the difference between visible and hidden nodes, call this dNeg
7. Add the values in dPos to their respective weights, and subtract the values in dNeg from their respective weights

Parallelizability

Activation of M nodes from N nodes is in serial O(M*N).

- Each node can activate itself independently, but depends on the value of N different nodes.
- Optimally, with M independent threads, time for activation would be O(N)

Accumulation of differences between M nodes and N nodes is, in serial, also O(M*N)

- However, each difference calculated depends on a constant number of values, and is independent of all the other differences calculated
- Optimally, with M*N independent threads, time for activation would be O(1)

Updating the weights is very similar to accumulation of differences, and can optimally run in O(1) time, given M*N threads.

Technology

SIMD parallel algorithm, or Single Instruction Multiple Data

GPUs are many, simple cores executing the exact same instructions on shared data.

OpenCL (Open Computing Language) is a relatively new standard that allows computation on a wide range of heterogeneous systems, including GPUs

Since the goal of this project was to speed up RBM-Provisor, I used JOCL, a thin JNI wrapper around OpenCL.

OpenCL

Basic Model of OpenCL:

- Central Command Queue, Host sticks work on, Compute units take work off and do it
- Kernels are pieces of code that are executed on the Compute units (GPU in this case)
- No Option Left Behind, resulting in library functions that take 7-10 arguments

RBM-Provisor

Has a class called LayeredRBM which is the deep belief network it uses.

Goal: Create a drop in replacement class that provides the same public functionality, parallel implementation through OpenCL.

Result: CLLayeredRbm, a deep belief network implemented using OpenCL for training and activation.

Benchmarking Results

Fortunately, there was a slight speedup. More fortunately, the code scales to run efficiently on pretty much anything.

Processor: Intel Core2Duo E6750 at 2.66GHz
Graphics Card: Nvidia Geforce 8800 GTX
(G80 GPU, 128 Cores, 575MHz core clock)

Trained on 4 children's songs

4.88 minutes to train with OpenCL
5.39 minutes to train with Original Code

Future Work

- Parallel Random Number Generation
  - Inherently hard, pseudo random number generators have a continuously updating state, each computation of the next random number depends on the previous.
- Tweak OpenCL synchronization:
  - Right now enqueues a barrier on the global work queue, essentially says "finish everything before the barrier before starting anything after"
- Global/Local workgroup sizes
  - Right now determined at runtime by OpenCL
  - Better choices could lead to more code actually being executed in parallel
- Copy multiple samples of data to GPU at same time
  - Data copying is expensive relative to computation
- Bells and Whistles